FINAL REPORT

NAG8-850

APRIL 1994

Stress Relaxation Functions

Methods of Approximation

by: Mark V. Bower, Ph. D., and Frederick S. Gant

Department of Mechanical and Aerospace Engineering
The University of Alabama in Huntsville

Stress Relaxation Functions

Methods of Approximation

by: Mark V. Bower, Ph. D., and Frederick S. Gant

Department of Mechanical and Aerospace Engineering
The University of Alabama in Huntsville

Table of Contents:

Executive Summary

Herein are reported two methods for determining the Prony series coefficients for viscoelastic stress relaxation data. Prony series are nonlinear functions of exponential terms and have a number of coefficients proportional to the number of terms. To fit a Prony series to stress relaxation data, the coefficients must be properly selected. Due to the nature of the exponential series and the number of terms, using methods such as least squares to determine coefficients requires that some coefficients must be estimated.

A new method of determining Prony series coefficients is presented. This method, the Domain of Influence Method, capitalizes on characteristics of the exponential decay curve to adjust its parameters to fit a set of data. In this development, this method was applied to viscoelastic stress relaxation data. The method is general and can be used to develop exponential decay curves to represent other types of data where appropriate.

The DOI method does not include any error correction within itself. To improve the results of the DOI method some form of error correction is necessary. As noted, the non linearity of the Prony series does not lend itself to common methods of error minimization. Optimization methods can be applied to this problem. These methods use the functional behavior of the problem under study to minimize or maximize some characteristic of the problem. Here the function minimized was an error function between the DOI estimated Prony series and the viscoelastic data. Optimization was achieved by adjusting the Prony series coefficients to minimize that error.

The DOI method was encoded in FORTRAN and integrated with commercially available optimization routines to produce a tool called Viscoelastic Coefficient Determination or VCD. A description of this code including a discussion of the salient features is presented. An example is used to demonstrate the DOI method, illustrate the operation of VCD, and demonstrate the capabilities of the method and the software. A code listing appears in the Appendix.

The authors believe that the DOI method coupled with error optimization is a powerful tool for accurate Prony series representation of viscoelastic data.

Introduction

Modeling and predicting the behavior of any material system requires knowledge of the mechanical behavior of the material. It would be preferable to test every conceivable loading condition a material may be subjected to and observe its performance, but this is not practical or possible. Instead, standard tests are conducted on a limited number of material samples and constitutive models, constructed to describe the behavior, are used with the test data to predict the behavior of the material for a wide range of loading conditions.

Solid rocket propellant typically consists of a binder with suspended particles, hence the term composite propellant. Thus, solid rocket propellant is a composite in the same sense that concrete is a composite, that is, it is a particulate reinforced composite whose mechanical properties are isotropic. The binder acts as the fuel for the chemical reaction and is typically an elastomer or polymer. The suspended particles include the oxidizer for the chemical reaction and other additives. The mechanical behavior of a composite propellant is a function of many factors of which

the individual behavior of the constituents is a significant contributor. The particles tend to be elastic in behavior, while the elastomer or polymer binder tends to be viscoelastic.

A characteristic of viscoelastic response is stress relaxation behavior which is the time dependent response of a material to a given constant strain. For the material investigated in this research, the stress relaxation behavior is dominated by the binder's mechanical behavior. That is, they exhibit time dependent loading response and 'relax' over time.

The primary goal of this research was to identify the best form of the stress relaxation function to model the behavior of solid rocket propellant. A secondary goal was develop a procedure to determine constants in the function from material property data.

Phenomenological Model

The stress relaxation behavior of the propellant is determined through stress relaxation tests. In associated research activities [1], test coupons of the propellant were loaded to specified strain levels, the strain level then held constant for the duration of each test, and the load measured for a given time period. The data produced is the basis for the construction of a phenomenological model of the material behavior.

The data for the material in question is monotonically decaying in time with a slope that is monotonically increasing in time. These are characteristic observations for typical viscoelastic materials. Common practice in engineering analysis of viscoelastic materials has included the use of exponential functions, individually or in series, to represent the time dependent moduli and compliances. Justification for this practice has been

established from mechanical analog [2] and functional approaches [3], [4] to viscoelasticity. In the mechanical analog approach, the exponent of the exponential function is directly related to the viscous element in the model. Micro-molecular studies of polymer behavior [5] have added credence to the mechanical analog approach, relating the mechanical elements to molecular processes. In the furtherance of the functional approach, Bernstein [4] developed a mathematical theory that proves any monotonically decreasing function can be represented by the summation of exponentially decaying terms.

Considering the foregoing, the model used in this research is a series of exponentially decaying terms and of the form:

$$G(t) = G_0 + \sum_{i=1}^{N} G_i \exp\left(-t/\tau_i\right). \tag{1}$$

Prony [6] introduced a method for determining coefficients of a series of exponential functions through a least-squares method. In honor of his pioneering work, series composed of exponential functions are now referred to as Prony series. More recently, the collocation method has been used to determine Prony series coefficients. Tschoegl [7] presents a succinct description of the collocation method using Schapery's method for estimating relaxation times, τ_i.

Both the least-squares and collocation methods require that the exponents of the exponential terms be known (assumed) a priori. Typically, the exponents have been assumed to be evenly spaced on the decades of a logarithmic time scale. This has led to complications and erroneous results in viscoelastic analyses.

A goal of this kind of research is to relate the specific coefficients and exponents to specific molecular material parameters similar to the results of Doi and Edward's research [5].

Domain of Influence

Background

As discussed in the previous section, the Prony series is a summation of decaying exponential functions. The behavior of these functions determines how well the Prony series represents a given set of stress relaxation data. As in the development of any numerical approximation, the unknown coefficients, and for a Prony series the exponents, in the approximating function must be determined in such a way that the error between the data and the approximation is minimized. Typically this is accomplished by use of a least-squares method. Unfortunately, due to the nature of the exponential function (the exponential terms do not form an orthogonal set, which is fundamental to the application of the least-squares method [8]) this process results in a system of highly nonlinear equations for the coefficients and exponents. To simplify this problem previous methods for determining values of G_i and τ_i have involved estimating a value

for τ_i, a priori, and determining G_i using a least-squares method. For the purposes of this discussion this method will be referred to as the "Time Constant Selection" method.

Due to the lack of suitability of the exponential terms for the least squares method and difficulties in estimating the time constants, it is not uncommon for the Prony series fit to oscillate or 'wiggle' through the data points. Wiggling results from having too few or too many exponential terms in the Prony series fit. In the event of too few terms, increasing the number of exponential terms can reduce or eliminate the wiggle. However, increasing the number of exponential terms increases the number of assumed constants (which may affect fit accuracy) and increases the required computational effort in the Time Constant Selection method.

Another method of fitting stress relaxation data involves plotting the logarithm of the modulus versus the logarithm of time. Often the resulting curve is a nearly straight line. The slope and intercept of this line may then be determined using a least-squares method. Limitations of this method include: the result is a single term Prony series, the approximation is not performed on the original data, and the accuracy of the approximation may be poor [9].

To improve the results obtained from the Time Constant Selection method it is necessary to improve the estimation of τ_i. To achieve this goal, it is useful to review the behavior of the decaying exponential functions. Figure 1 shows several decaying exponential functions with different time constants plotted versus the logarithm (base 10) of time. These curves are generated from:

$$\gamma(t) = \gamma \cdot e^{\frac{-t}{\tau}} \tag{2}$$

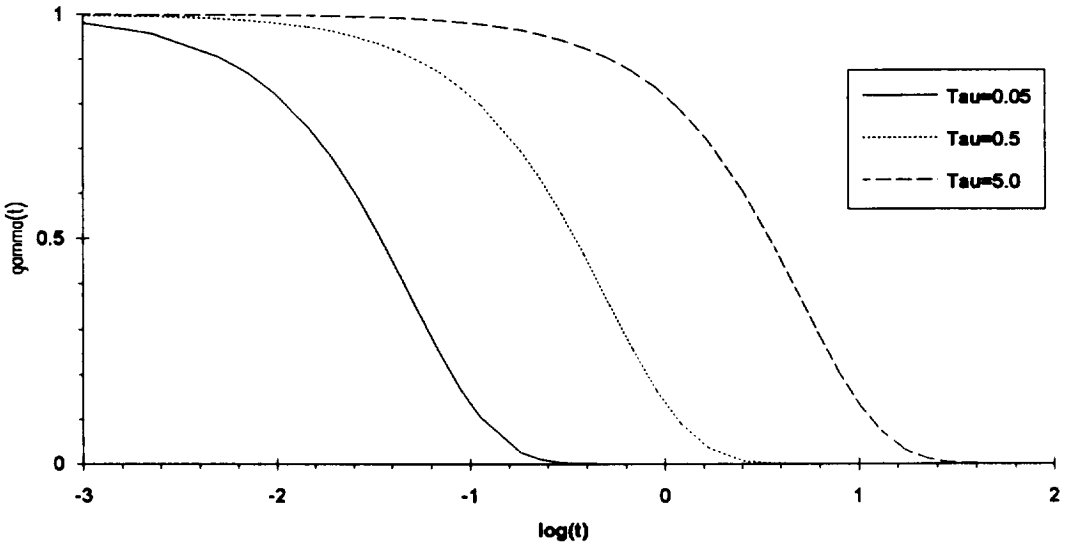

Figure 1 -- Effect of τ on Exponential Terms.

Figure 1 shows the effect of the exponent in an exponential function. Note that γ is 1.0. This figure demonstrates that: (1) changes in the exponent of an exponential function produce a shift of the curve along the time axis; (2) the majority of the decay takes place over a limited range of time; and (3) in this range the curve's slope (that is, the rate of change relative to the logarithm of time) is non-zero and near zero outside this range. Solving equation (2) for t yields:

$$t = -\tau \cdot \ln\left(\frac{\gamma(t)}{\gamma}\right). \tag{3}$$

From this equation a quantity, $\delta\log(t)$ can be defined that gives the number of decades over which a given percentage of $\gamma(t)$ decays. Recalling that γ=1 in equation (3), we may write:

$$t = -\tau \cdot \ln(\gamma(t)). \tag{4}$$

Now define the value of $\gamma(t) = a$ subject to the constraint: $0.5 < a < 1.0$. From the value of 'a' define: $b = 1 - a$ and $c = 1 - b = 1 - 2a$ (for example: if $a = 0.05$, $b = 0.95$, and $c = 0.90$). Recognize that 'a' and 'b' will have times, t_a and t_b associated with them through equation (4):

$$t_a = -\tau \ln(a) \text{ and } t_b = -\tau \ln(b).$$

Now define $\delta\log(t)$ as:

$$\delta \log(t) = \log(t_a) - \log(t_b) = \log(-\tau \ln(a)) - \log(-\tau \ln(b)) \tag{5}$$

Simplifying equation (5) yields:

$$\delta \log(t) = \log\left(\frac{\ln(\gamma(t))}{\ln(1-\gamma(t))}\right). \tag{6}$$

From equation (6) and using $\gamma(t) = 0.05$, one finds that $\delta\log(t)$ is 1.77. Expressed in another way, 90 percent of the decay takes place in 1.77 decades of time. This region is defined as the "domain of influence" for this exponential function.

Figure 2 is shown to develop a better understanding of the limited range of non-zero slope, i.e., the domain of influence. This figure presents a plot of the slope of the exponential function shown in Figure 1 versus the logarithm of time. Note, these derivatives are not with respect to time, but, rather with respect to the logarithm of time. The derivatives with respect to the logarithm of time are:

$$\frac{d\gamma(t)}{d\log(t)} = \frac{-t}{\tau} \cdot \ln(10) \cdot e^{\frac{-t}{\tau}} \tag{7}$$

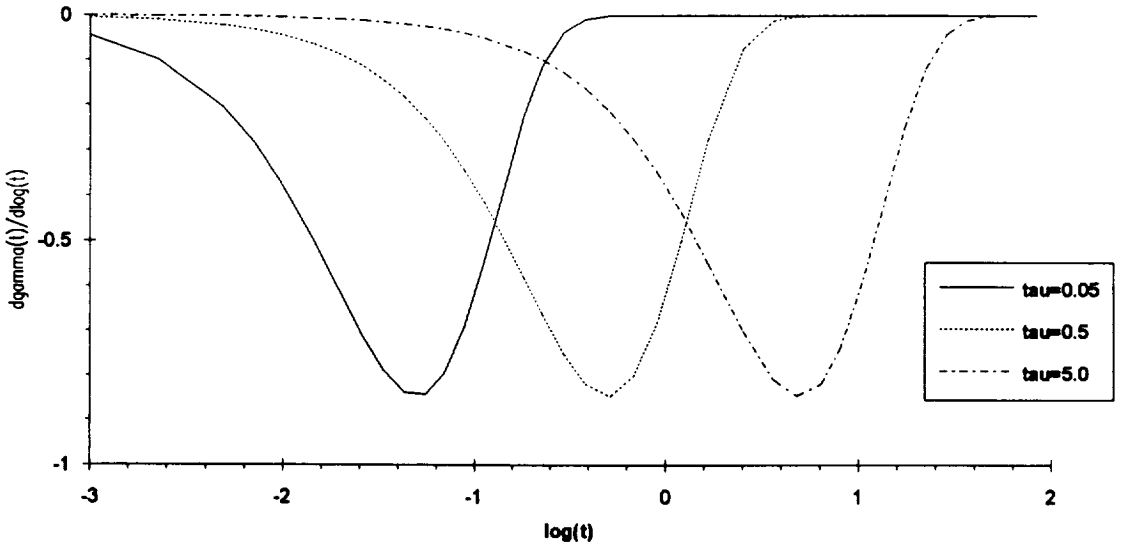

Figure 2 -- Slope of Exponential Function.

The conclusions of this study of the effects of the time constants are: (1) an exponential term decays, i.e., has non-zero slope, over a limited range of time, a limited domain of influence; and (2) the value of τ locates the peak of the slope curves and the range of the exponential decay. These conclusions agree with the conventional wisdom from control system behavior that an exponentially decaying signal or output will decay to less than one percent of its original value in a time equal to three time constants (the exponent).

Figure 3 shows a plot of an exponential decay term for various values of γ in equation (2), note that here τ is constant.

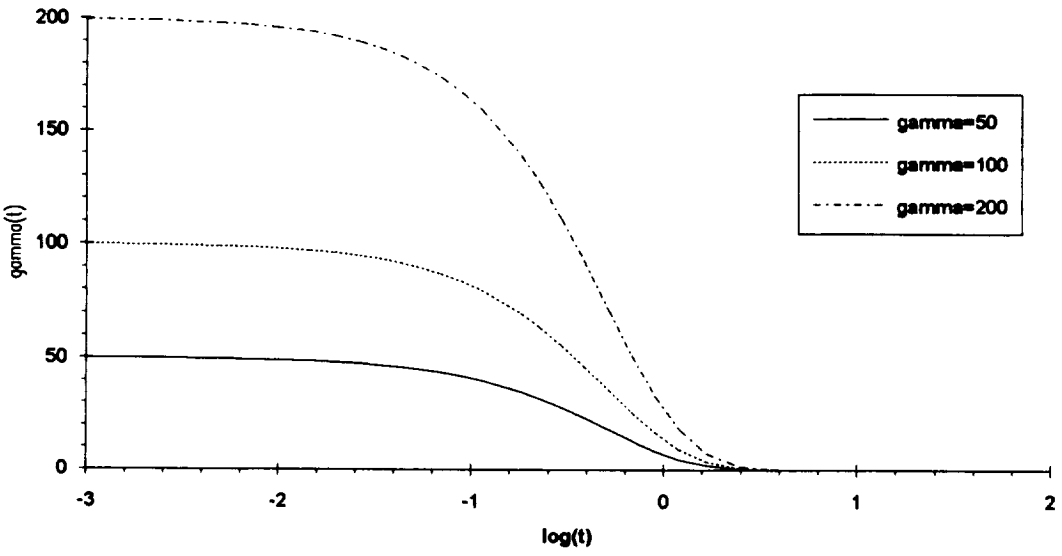

Figure 3 - Effects of γ on Exponential Terms

The conclusion from this study is that the γ terms determine the range on the modulus axis over which a particular exponential term will act.

Method

Knowing the characteristics of the exponential function, a numerical procedure can be devised that gives good agreement between a stress relaxation data set and the Prony series. There are several steps to this process. Table 1 shows an example of this method as applied to a portion of a stress relaxation data set. The procedure is as follows:

Step 1

Define the interval over which the exponential terms operate in terms of decades. This is analogous to the $\delta\log(t)$ of equation (4). In the example in table 1, $\delta\log(t)=0.80$.

Step 2

Define the change in log(t) as:

$$\delta \log(t_i) = \log(t_i) - \log(t_1)$$

where t_i is a time point in the stress relaxation data set and t_1 is the first time point.

Step 3

Compute the $\delta\log(t_i)$ for the time data of the data set.

Step 4

Review the $\delta\log(t_i)$ values to find the data point, i, where the value exceeds the interval limit set in step 1.

Step 5

Repeat steps 3 and 4 using the i+1 data point as the first data point in equation (5). Repeat this step until the time data set is divided into N groups.

Step 6

Divide the modulus (G) data into groups according to the divisions defined in the time data.

Step 7

Normalize the modulus (G) data using:

$$\text{Norm}(G_i) = \frac{G_i - G_n}{G_1 - G_n}$$

where G_i is the i^{th} G data point, G_1 is the first modulus (G) data point in a given data group, and G_n is the last modulus (G) data point in a given data group.

Step 8

Find the τ_i by using linear interpolation. Identify data point k and k+1 such that e^{-1} (0.3676) lies between the normalized values of G_k and G_{k+1}. The exponent τ_i is found from:

$$\tau_i = \left(e^{-1} - \delta\ln(G_k)\right)\frac{t_{k+1} - t_k}{\delta\ln(G_{k+1}) - \delta\ln(G_k)} + t_k$$

In table 1, the last column contains the τ values. Note that these values are associated with the first of the two scaled del G(t) that bracket the value of e^{-1}.

Step 9

Modulus values, that is the G's, are the first values of G in the N groups of G values. These are the boxed values in the del G(t) column of table 1.

Step 10

The steady state modulus value is the mean value of the last three G data values.

Step 11

Scale the values of G_i using the values of the exponential and the associated τ_i, to wit:

$$\tilde{G}_i = \frac{G_i}{\exp\left(-t_1/\tau_i\right)}$$

The N exponents and N coefficients determined from the above procedure are the required Prony series exponents and coefficients for the stress relaxation modulus function.

Comments on the Method

In step 10, the steady state, that is, elastic modulus is the average of the last three G data points. This requires that the data reach a steady state (or at least not change significantly) for at least those three points.

Through step 10 we have ignored the influence of the exponential components of equation (1) upon the G_i modulus terms. Note that the sum of G_o and the G_i's is equal to the first G data point, that is:

$$G_{data_1} = G_o + \sum_{i=1}^{N} G_i$$

Comparing the above equation with equation (1), evaluated at the first time point using the coefficients and exponents determined through step 10, it is clear that the value of $G(t_{data_1})$ will not equal G_{data_1} due to the value of the

exponential. For the data considered and the equations described, the exponential will always have a value between 1 and 0 (see figure 1). Also the value of: $\exp\left(-t_{data_1}/\tau_i\right)$ for τ_i in the range: $t_{data_1} < \tau_i < 10 * t_{data_1}$ will not be close to one. This results in a Prony series fit which does not match the data for the first part of the curve, specifically, the fit values are lower than the data. An example of this result is presented in figure 4.

To account for the influence of the exponential it is necessary to set the Prony series fit equal to G_{data_1} and scale the G_i coefficients:

$$G_{data_1} = G_o + \sum_{i=1}^{N} G_i = G_o + \sum_{i=1}^{N} \tilde{G}_i \exp\left(-t_{data_1}/\tau_i\right)$$

where $\tilde{G}_i$ are the scaled coefficients. One possible solution to the above equation is to set the product of the scale coefficients and the exponential equal to the coefficient determined in step 9. The equation of step 11 results from this solution. Applying these scale factors to the DOI fit of figure 4 gives the curve presented in figure 5.

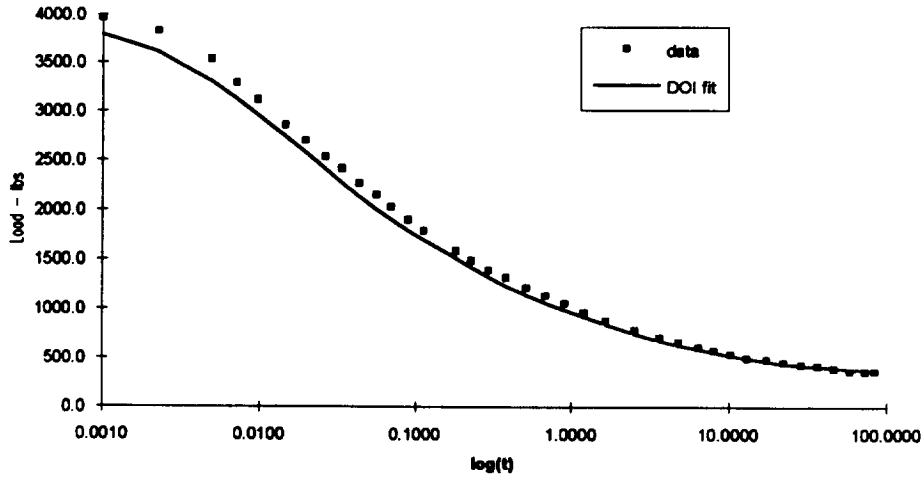

Figure 4. DOI Results for Data in Table 1.

Table 1. An Example of the Domain of Influence Method

Time (min)	log(t)	del log(t)	G(t) Data	del G(t)	scaled del G(t)	τ
0.0010	-3.0000	0.0000	3964.5	673.0	1.0000	
0.0023	-2.6469	0.3531	3823.4	531.9	0.7904	0.0049
0.0049	-2.3083	0.6917	3536.3	244.8	0.3638	
0.0071	-2.1472	0.0000	3291.5	1132.5	1.0000	
0.0097	-2.0144	0.1328	3111.7	952.7	0.8413	
0.0146	-1.8364	0.3108	2861.3	702.3	0.6201	
0.0195	-1.7093	0.4379	2701.7	542.7	0.4792	0.0248
0.0264	-1.5788	0.5684	2537.7	378.7	0.3344	
0.0334	-1.4762	0.6710	2418.4	259.3	0.2290	
0.0432	-1.3644	0.7829	2274.9	115.9	0.1023	
0.0555	-1.2559	0.0000	2159.0	839.4	1.0000	
0.0695	-1.1580	0.0979	2033.2	713.6	0.8501	
0.0890	-1.0507	0.2053	1905.5	585.8	0.6979	
0.1126	-0.9484	0.3075	1793.4	473.8	0.5645	0.1673
0.1809	-0.7425	0.5134	1587.5	267.8	0.3191	
0.2265	-0.6450	0.6110	1489.5	169.9	0.2024	
0.2938	-0.5319	0.7240	1397.2	77.6	0.0924	
0.3828	-0.4170	0.0000	1319.6	536.6	1.0000	
0.5160	-0.2873	0.1297	1218.1	435.0	0.8107	
0.6818	-0.1664	0.2506	1135.0	352.0	0.6559	
0.9021	-0.0447	0.3723	1057.5	274.5	0.5115	1.1410
1.1995	0.0790	0.4960	961.6	178.6	0.3327	
1.6607	0.2203	0.6373	882.4	99.3	0.1851	
2.5274	0.4027	0.0000	783.0	304.7	1.0000	
3.6455	0.5618	0.1591	707.8	229.5	0.7532	
4.7850	0.6799	0.2772	660.4	182.0	0.5975	
6.3953	0.8059	0.4032	605.8	127.4	0.4182	7.1208
8.0163	0.9040	0.5013	571.5	93.1	0.3057	
10.2722	1.0117	0.6090	539.4	61.0	0.2004	
12.9772	1.1132	0.7105	498.3	19.9	0.0654	
17.3240	1.2386	0.0000	478.4	117.0	1.0000	
22.2840	1.3480	0.1093	451.3	90.0	0.7691	
28.7547	1.4587	0.2201	430.7	69.3	0.5922	
36.4954	1.5622	0.3236	418.0	56.6	0.4840	41.5427
46.2945	1.6655	0.4269	391.6	30.2	0.2585	
58.3599	1.7661	0.5275	362.7	1.3	0.0115	
73.6437	1.8671	0.6285	357.6	-3.7	-0.0319	
84.5043	1.9269	0.6882	361.4	0.0	0.0000	

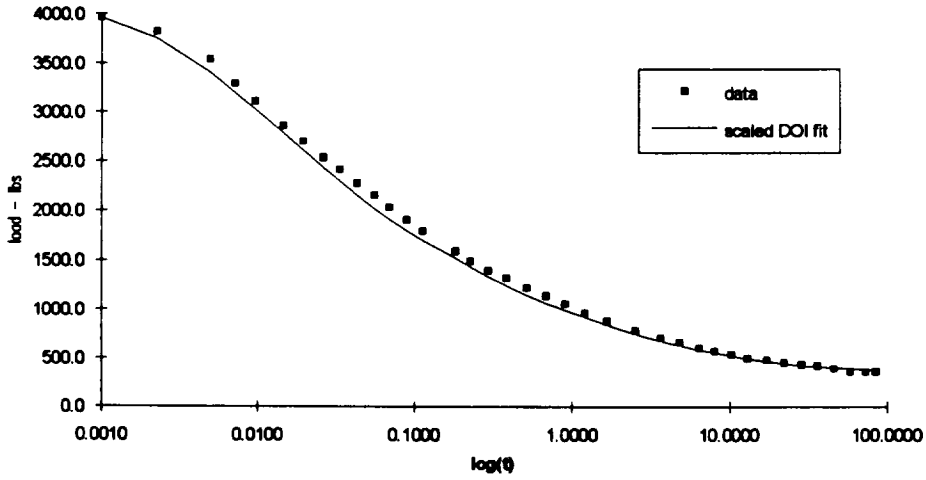

Figure 5. Scaled DOI Results in Table 1.

This is a simple procedure for determining the exponents and coefficients of a Prony series approximation of stress relaxation data based on the domain of influence of the individual exponential functions. This procedure shows good correlation with test data and superior performance in comparison to other methods. The procedure is easy to implement by hand or using spreadsheet programs, such as: Microsoft® Excel or Lotus® 1-2-3. Apparently, this method can determine the Prony series term exponents and coefficients for any data set that has monotonically decreasing absissae with increasing ordinate.

It is important to note that the DOI method is not limited by a large number of data points and is not sensitive to irregularly spaced data. However, due to the linear interpolation used to find the exponent values, it is necessary to have at least three data points within any given data group.

A limitation of the DOI method is that there is no minimization of error between the Prony series approximation and the data.

Evaluating the error and revising the approximation based on that evaluation will lead to better agreement between the data and the Prony series. As noted before, due to the non linearity of the Prony series, simple, linear methods are not applicable for error correction in this problem. A more sophisticated approach must be used which is compatible with nonlinear equations. Numerical optimization is a nonlinear solution method applicable to error minimization.

Optimization Method

The Domain of Influence method provides a quick and effective determination of the Prony series coefficients. The results are the product of a single analysis of the data, which does not include any error evaluation, or minimization. This may be sufficient for some applications, however, it may not be sufficient for others. In these other applications it is desirable to evaluate the deviation of the DOI results from the given data. The deviation provides a measure of the quality of the curve fit.

Due to the lack of error evaluation in the DOI method, an additional operation must be performed if any improvement of the Prony series fit to the data is to be realized. As previously discussed, due to the non-linearity of the Prony series, standard techniques such as the least squares method will not work. Optimization techniques provide alternate methods for non-linear problems. Optimization methods, or "non-linear programming" methods, were applied to the problem of improving the accuracy of the Prony series coefficients and exponents determined by the DOI method. As is the case with a number of mathematical techniques,

optimization is an iterative procedure that is suitable for numerical implementation using computers and a variety of programming languages.

The general optimization problem statement is to minimize or maximize an objective function subject to constraints on the problem and limits on the independent variables. In the general optimization problem, the objective function describes some characteristic of the problem that is minimized or maximized (e.g., material used, cost, stress, etc.). The constraints describe some characteristic of the problem, i.e., a secondary dependent variable, which is not allowed to exceed or fall below a certain value (e.g., size, quantity, deflection, etc.). The limits on the independent variables of the design are self explanatory. [10]

The optimization problem for improving the fit of a Prony series to stress relaxation data is defined using an error function as the objective function. The error function used in the problem definition is:

$$\text{error} = \sqrt{\sum_{i=1}^{N}\left(\frac{G_{data_i} - G_{Prony}\left(t_{data_i}\right)}{G_{data_i}}\right)^2} \tag{8}$$

where G_{data_i} is the modulus data, t_{data_i} is the time data at some point i, G_{Prony} is the Prony series evaluated at that point in time, and the error is the square root of the summation of the squares of the normalized difference between the data and the Prony series fit for all data points in the set. In the objective function, G_{Prony} is determined using the Prony series, which is a function of the coefficients, G_k and τ_k, and the time value. The error function is minimized using an optimization technique which manipulates the objective function variables, that is, the Prony series coefficients.

For this optimization problem, no constraints were imposed, i.e., there are no other dependent variables with external restrictions. Limits for the Prony series coefficients and exponents, i.e., the independent variables, were established such that the G_k's and τ_k's could approach zero and would not become so large as to cause machine errors in calculation. In the formulation used, neither the coefficients nor exponents are permitted to be negative.

As might be expected from the form of the error function, sections of closely spaced data (relative to other sections of the data) weight the error function. When optimizing a data set with an area of high data density, the optimization favors the high density area. Consequently, to obtained the best results the data must be evenly spaced on a logarithmic scale.

Optimization of the coefficients and exponents determined from the DOI method was performed using Design Optimization Tools (DOT) software from VMA Engineering. These are a set of FORTRAN routines that contain algorithms to perform numerical optimization and can be applied to most applications wherein optimization is appropriate. For unconstrained non-linear optimization DOT uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm. This method uses the objective function to define a "search" direction from the initial values of the function using the gradient of the function. In particular, the initial "search" direction, i.e., direction in which the independent variables are incremented, is selected by determining the direction of steepest descent. Subsequent search directions are computed using the product of an estimate of the inverse of the Hessian matrix and the gradient of the

objective function. The estimate of the inverse Hessian contains information about the previous search direction which improves the convergence to an optimal solution [11].

The BFGS method, like all optimization methods, may yield several "solutions" due to the presence of local minima in the objective function [12]. To find the global minimum, that is, the lowest value of the objective function, and, consequently for this application, the least deviation from the data, it is necessary to begin the search in close proximity to this optimum. This requires that the initial values of the independent variables be close to the optimal values. The DOI method provides such an initial guess.

Paring the DOI method of Prony series estimation and the optimization techniques yields more accurate series coefficients which better represent the data. This is reflected in a reduction of the error function as determined by equation (8).

Viscoelastic Coefficient Determination Program

Program Documentation

The DOI method and Prony series coefficient optimization have been implemented in a program called Viscoelastic Coefficient Determination or VCD. This program is a collection of routines written in FORTRAN which numerically implement the DOI method and use the DOT optimization routines to refine the Prony series coefficients.

The program VCD is fairly simple and efforts were made during its development to adequately comment the code. Generally, VCD is built in a modular fashion making use of subroutines and functions to perform specific tasks. Variables used within the program by many of the subroutines are passed about in a common block called DATA. Figure 6 gives a flow diagram illustrating the salient features of VCD.

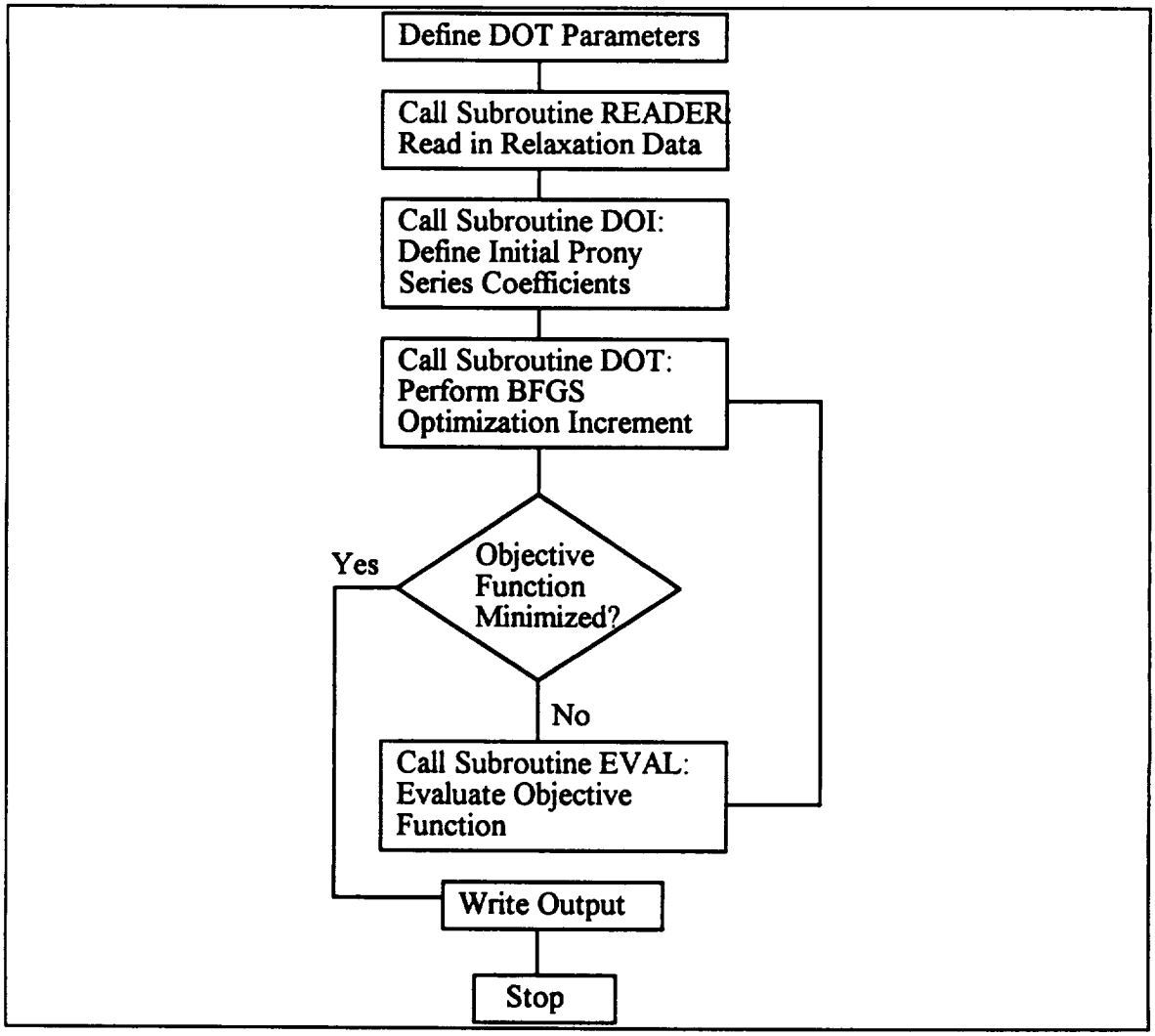

Figure 5. Flow Chart of Program VCD

VCD is constructed as follows:

Step 1

A set of variables are initialized which establish limits on the optimization input variables and the optimization is defined to be unconstrained optimization. The setting for unconstrained optimization invokes the BFGS method of optimization.

Step 2

The program then loads data from an input file.

Step 3

The routine which reads the input file, subroutine READER, first prompts the user for an input file name. The input file must have the format: first line is the number of data points and following lines are data points the first value being the time and the second value is the modulus value. Tab delimited data is acceptable.

Step 4

After the data has been loaded into the data arrays in VCD, the DOI method is used to determine the number of terms and the initial values for G and τ in the Prony series.

Step 5

Subroutine DOI implements the DOI method using the steps outlined in the METHOD section of this report. This particular implementation takes the average of the last 3 modulus values as the steady state or elastic modulus for the material. This is based on the assumption that the data achieves a steady state value for at least the last 3 points.

Step 6

The DOT implementation of the BFGS optimization method uses the DOI results as the initial Prony series coefficients. DOT optimizes using an iterative procedure which manipulates the Prony series coefficients and passes those coefficients out of the program to be used in the objective function evaluation. Objective function evaluation, that is data/fit error, is done in subroutine EVAL and the results returned to DOT. The results of the objective function evaluation are compared to optimization solution convergence parameters to determine if the error has been sufficiently minimized.

The VCD input file must have the format presented in Table 2.

Table 2. Sample VCD Input File

38	
0.0010	3964.5
0.0023	3823.4
0.0049	3536.3
0.0071	3291.5
0.0097	3111.7
0.0146	2861.3
0.0195	2701.7
0.0264	2537.7
0.0334	2418.4
0.0432	2274.9
0.0555	2159.0
:	:
:	:
84.5043	361.4

Once an input file has been generated that conforms to the above format, VCD can be run using that file as input. The compiled and linked FORTRAN VCD code is run by typing VCD at the DOS prompt.

VCD first prompts the user for the name of the input file, here it is important to note that the file name must be enclosed in single quotes, for example 'MYFILE.INP'. Using that file for the DOI process and the optimization operation, VCD performs these operations without any user intervention. As it loops through the optimization loop it writes the statement, "working . . ." on the screen once per loop. When VCD successfully ends its execution the statement, "Stop - Program terminated." is written to the screen.

Once VCD completes its optimization the DOT subroutines generate an output file that reports on DOT's actions during the optimization process; this file is called "OUTPUT". VCD produces an output file named "COEFF.OUT" which contains the initial set of Prony series coefficients from DOI and the optimized coefficient set from the DOT routines. The file "COEFF.OUT" is an ASCII text file that can be imported into other programs or software for further analysis or study. The contents of the data file have the following format and form:

```
number of terms:                3
DOI Results
        410.3060000
        123.0993000      5.177419E-001
         93.2304100          3.6075040
         15.8109600          8.7106520

Optimized Result
        375.4074000
        115.3154000      4.676060E-001
        111.6172000          6.2780050
         30.5934300          6.2728110
```

The coefficients in the two blocks are arranged as follows:

G_0
G_1 t_1
G_2 t_2
: :
G_n t_n

References

1. Little, R. R., "NASA SPIP Program Quarterly Status Report, 4th qtr. FY 91", Letter Report RD-PR-91-54, Nov. 1991.
2. Flugge, W., "Viscoelasticity", 2nd ed., Springer Verlag, Berlin, 1975.
3. Staverman, A. J. and Schwarzal, F., "Nonlinear Deformation Behavior of High Polymers", Physik der Hochpolymeren
4. Bernstein, S., "Sur Les Fonctions Absolument Monotones", Acta Mathematica, vol. 52, 1928, pgs. 1-66.
5. Doi, M. and Edwards, S. F., "Dynamics of Concentrated Polymer Systems, Part 3. - The Constitutive Equation", J. C. S. Faraday II, vol. 74, 1978, pgs. 1818-1832.
6. Buckingham, R. A., "Numerical Methods", Sir Isaac Pitman and Sons, Ltd., London, 1957, pgs. 329-333.
7. Tschoegl, Nicholas W., "The Phenomeological Theory of Linear Viscoelastic Behavior", Springer-Verlag, Berlin Heidelberg New York, 1989, pgs. 137-143.
8. Burden, R. L. and Faires, J. D., "Numerical Analysis", 3rd ed., Prindle, Weber and Schmidt, Boston, 1985.
9. Little, R. R., "U. S. Army Missle Command Quarterly Status Report Second Quarter, FY 92", Letter Report RD-PR-92-24, Mar. 1992.
10. Vanderplaats, G. N. and Hansen, S. R., "DOT User's Manual", VMA Engineering, Goleta, CA, 1990.

11. Vanderplaats, G. N., "Numerical Optimization Techniques for Engineering Design: With Applications", McGraw Hill, New York, 1984.

12. Vanderplaats, G. N. and Hansen, S. R., "DOT User's Manual", VMA Engineering, Goleta, CA, 1990, pg. E-9.

Acknowledgments:

This research was performed under NASA contract NAG-850.

Appendices

Contained herein are the code listing of the VCD program. This listing does not include the DOT source code as that is a large body of code which is not easily interpreted. Also due to the relative robustness of the DOT code, detailed knowledge of its internal structure and function are not necessary to exercise the code's utility. The DOT code is invoked by the one subroutine call to DOT in program VCD.

Contents

File VCD

```
      1 $DEBUG
      2       PROGRAM VCD
      3 C
      4 C Program to use the DOT subroutine for finding Prony series
      5 C coefficients to fit viscoelastic material test data.
      6 C
      7       DIMENSION X(40),XL(40),XU(40),G(20),
      8      *WK(1400),IWK(200),RPRM(20),IPRM(20)
      9       COMMON/DATA/ N, NTERM, EDATA(200), TDATA(200)
     10
     11 C DEFINE NRWK, NRIWK
     12       NRWK=1400
     13       NRIWK=200
     14
     15 C ZERO RPRM AND IPRM
     16       DO 10 I=1,20
1    17         RPRM(I)=0.0
1    18   10    IPRM(I)=0
     19
     20 C Override the default for CTMIN which is the minmum value
     21 C of a constraint (essentially program zero)
     22 C      RPRM(2) = 0.005
     23       RPRM(3) = 0.0001
     24       RPRM(4) = 0.000001
     25
     26 C override the default for ITMAX, the max number of iterations
     27       IPRM(2) = -1
     28       IPRM(3) = 1000
     29       IPRM(4) = 10
     30
     31 C DEFINE METHOD
     32       METHOD=1
     33
     34 C
     35 C DEFINE IPRINT, MINMAX, INFO
     36 C
     37       IPRINT = 3
     38       MINMAX = -1
     39       INFO = 0
     40
     41 C  Name the DOT output file
     42       open(6, file='output', status='new')
     43
     44 C
     45 C Subroutine READER reads data to be used in the test
     46 C INFO is passed as a dummy varible and is not used
     47 C
     48       CALL Reader(INFO)
     49
     50 C Open a file for output of the DOI results and the
     51 C optimization results
     52       Open(9,file='coeff.out',status='new')
     53 C
     54 C Subroutine DOI reviews the data and make a guess of how many
     55 C terms to use, NTERM, and values for E(i) and tau(i)
     56 C
     57       CALL DOI(X)
```

```
      58
      59 C Write the DOI results to the output file
      60       write(9,*) 'number of terms:  ', nterm
      61       write(9,*) 'DOI Results'
      62       write(9,*) X(1)
      63       do 15 i = 1, nterm
1     64  15    write(9,*) X(2*i), X(2*i+1)
      65
      66 C Define the number of design variables and constraints
      67 C since this is an unconstrained problem NCON=0
      68       NDV = 1 + 2 * NTERM
      69       NCON = 0
      70 C
      71 C DEFINE BOUNDS, these are side constraints
      72 C
      73       DO 20 I=1,NDV
1     74         XL(I) = 1.0E-10
1     75    20   XU(I) = 1.0E20
      76
      77
      78 C
      79 C DOT LOOP - DOT optimizes the values of the X array.  In this
      80 C            application, the X array contains the values of the
      81 C            G and tau viscoelastic coefficients.
      82 C
      83  100  CALL DOT(INFO,METHOD,IPRINT,NDV,NCON,X,XL,XU,
      84      *OBJ,MINMAX,G,RPRM,IPRM,WK,NRWK,IWK,NRIWK)
      85       IF(INFO.EQ.0) GOTO 110
      86 C
      87 C Subroutine EVAL evaluates the objective function and returns its
      88 C value in OBJ.  It also evaluates the value of the constraints and
      89 C returns those values in the array G.
      90 C
      91       CALL EVAL(OBJ,X,G,NTERM)
      92       Write(*,*) 'working . . . '
      93       GO TO 100
      94 C
      95 C Once DOT is satisfied, write out the Prony series
      96 C coefficients for the user's amusement
      97 C
      98  110  continue
      99
     100 C Write the results of the optimization to the output file
     101       write(9,*) ' '
     102       write(9,*) 'Optimized Result'
     103       write(9,*) X(1)
     104       do 120 i = 1, nterm
1    105  120   write(9,*) X(2*i), X(2*i+1)
     106       STOP
     107       END

Name    Type         Offset P Class

EDATA   REAL              8   /DATA  /
G       REAL            496
I       INTEGER*4      7144
INFO    INTEGER*4      7160
IPRINT  INTEGER*4      7152
IPRM    INTEGER*4      7056
IWK     INTEGER*4      6176
```

```
METHOD  INTEGER*4       7148
MINMAX  INTEGER*4       7156
N       INTEGER*4          0   /DATA  /
NCON    INTEGER*4       7172
NDV     INTEGER*4       7168
NRIWK   INTEGER*4       7140
NRWK    INTEGER*4       7136
NTERM   INTEGER*4          4   /DATA  /
OBJ     REAL            7180
RPRM    REAL            6976
TDATA   REAL             808   /DATA  /
WK      REAL             576
X       REAL              16
XL      REAL             176
XU      REAL             336


Name     Type           Size   Class

DATA                    1608   COMMON
DOT                            SUBROUTINE
EVAL                           SUBROUTINE
DOI                            SUBROUTINE
READER                         SUBROUTINE
VCD                            PROGRAM

 Pass One     No Errors Detected
             107 Source Lines
```

File READ

```
      1 $DEBUG
      2       Subroutine Reader(d)
      3       Integer d, n
      4       character*8 fn
      5       COMMON/DATA/ N, NTERM, E(200), T(200)
      6 C
      7 C This subroutine acquires the viscoelastic test data from a file
      8 C and puts it into the E and T arrays of the common block data.
      9 C
     10 C First read the name of the data file from the keyboard
     11 C
     12       Write(*,*) 'Please give me the name of the data file'
     13       Read(*,*) fn
     14       Open(7,file=fn,status='old')
     15 C
     16 C Read the number of data points
     17 C
     18       Read(7,*) n
     19       write(*,*) 'n', n
     20 C
     21 C For the number of data points, read data from the file
     22 C
     23       do 10 I=1,N
1    24   10    read(7,*) T(I), E(I)
     25
     26       return
     27       end

Name     Type         Offset P Class

D        INTEGER*4         0 *
E        REAL              8   /DATA  /
FN       CHAR*8           16
I        INTEGER*4        24
N        INTEGER*4         0   /DATA  /
NTERM    INTEGER*4         4   /DATA  /
T        REAL            808   /DATA  /


     28

Name     Type           Size   Class

DATA                    1608   COMMON
READER                         SUBROUTINE


 Pass One    No Errors Detected
             28 Source Lines
```

File DOI

```
      2         Subroutine DOI(x)
      3 C
      4 C  This program evaluates the data and makes a good guess
      5 C  for the number of coefficients necessary, nterm, and
      6 C  the values E and tau
      7 C
      8       DIMENSION x(40), Ei(40), tau(40), ndx(20)
      9       Real delt, delE(200), sp
     10       integer j, i
     11
     12       COMMON/DATA/ N, NTERM, E(200), T(200)
     13
     14 C
     15 C Begin searching the time data for operational ranges of the
     16 C exponential function.  This will detemine where in the data
     17 C an exponential term will operate and provide the locations,
     18 C that is the number of the element, in the data set in the
     19 C index array ( ndx(j) )
     20 C
     21 C Initilize the counter varible j and the first ndx element
     22 C j will be used to increment the ndx array
     23 C
     24       j = 1
     25       ndx(1) = 1
     26 C
     27 C the conditional checks to see if the delta of log t is less
     28 C than the 0.80 decade operational range of the exponential function
     29 C if it is then increment the j counter and record where that occured
     30 C in the ndx array
     31 C
     32       do 20 i = 1, N
1    33         delt = log10(t(i)) - log10(t(ndx(j)))
1    34         if (delt .lt. 0.80) goto 20
1    35         j = j + 1
1    36         ndx(j) = i
1    37   20  continue
     38 C
     39 C Record how many sets within the data were found
     40 C
     41       nterm = j
     42 C
     43 C If the last index position is the last data point then there is
     44 C one fewer sets within the data
     45 C
     46       if(ndx(j) .eq. N) nterm = j - 1
     47 C
     48 C Define the last ndx term to be the last data point
     49 C
     50       ndx(j+1) = N
     51 C
     52 C Now work on the E values to find a normalized value of
     53 C the E values in the sets within the data
     54 C
     55       j = 2
     56
     57       do 40 i = 1, N-1
1    58         if (i .eq. ndx(j)) j = j + 1
```

```
1     59 C        write(*,*) 'j=',j, '   i=',i
1     60 C        write(*,*) 'E(',ndx(j-1),')=',E(ndx(j-1)),E(ndx(j))
1     61         delE(i) = (E(i)-E(ndx(j))) / (E(ndx(j-1))-E(ndx(j)))
1     62    40  continue
      63
      64        delE(N) = 0.0
      65
      66 C
      67 C The delE array is the delta in the E values broken into sets
      68 C associated with the 0.80 decade time ranges.  In each set, delE
      69 C ranges from 1 to 0.
      70
      71
      72 C Now find the tau's associated with the sets of delE data that are
      73 C defined by the ndx array.
      74 C
      75 C  initialized the counter varibles
      76 C
      77        j = 1
      78        i = 0
      79 C
      80 C Begin the loop.  Loop will find the points in the data
      81 C set between which the value of exp(-1) lie and linearly
      82 C interpolate what a guess for tau should be.
      83 C
      84        sp = exp(-1.0)
      85
      86    50  i = i + 1
      87          if (delE(i) .gt. sp) goto 60
      88          tau(j) = ylntrp(sp,t(i-1),delE(i-1),t(i),delE(i))
      89          j = j + 1
      90          i = ndx(j)
      91    60  if (i .lt. N) goto 50
      92 C
      93 C Find the Ei values associated with the sets in the data
      94 C
      95        do 70 k = 1, nterm
1     96     70    Ei(k) = E(ndx(k)) - E(ndx(k+1))
      97
      98 C
      99 C Now condition the E data to account for the effects of the
     100 C exponential terms using the taus defined
     101 C
     102        do 80 k = 1, nterm
1    103 C         write(*,*) Ei(k), T(1), tau(k)
1    104    80    Ei(k) = Ei(k)/exp(-T(1)/tau(k))
     105 C
     106 C Define the constant term of the coefficient set Eo as the
     107 C average of the last three E(i) data points and make this
     108 C the first element of the x array
     109 C
     110        x(1) = (E(N-2) + E(N-1) + E(N)) / 3.0
     111 C
     112 C Put the results into the x array for DOT to use
     113 C
     114        do 90 k = 1, nterm
1    115 C         write(*,*) 'Ei(',k,')',Ei(k), ' tau(',k,')',tau(k)
1    116          x(k * 2) = Ei(k)
1    117    90    x(k * 2 + 1) = tau(k)
     118
```

```
    119       return
    120
    121       Stop
    122       End

Name    Type         Offset P Class

DELE    REAL            416
DELT    REAL           1228
E       REAL              8   /DATA  /
EI      REAL             16
EXP                          INTRINSIC
I       INTEGER*4      1220
J       INTEGER*4      1216
K       INTEGER*4      1240
LOG10                        INTRINSIC
N       INTEGER*4         0   /DATA  /
NDX     INTEGER*4       336
NTERM   INTEGER*4         4   /DATA  /
SP      REAL           1236
T       REAL            808   /DATA  /
TAU     REAL            176
X       REAL              0 *
YLNTRP REAL                  FUNCTION


    123
    124
    125       Function ylntrp(y,x1,y1,x2,y2)
    126 C
    127 C This function returns the linear interpolation of x at
    128 C the given value of y between the points x1,y1 and x2,y2
    129 C
    130       real x, y, x1, y1, x2, y2, ylntrp
    131       x = ((x2 - x1) * (y - y1)) / (y2 - y1) + x1
    132       ylntrp = x
    133
    134       return
    135       stop
    136       end

Name    Type         Offset P Class

X       REAL           1256
X1      REAL              4 *
X2      REAL             12 *
Y       REAL              0 *
Y1      REAL              8 *
Y2      REAL             16 *
    137
    138

Name    Type           Size   Class

DATA                   1608   COMMON
DOI                           SUBROUTINE
YLNTRP REAL                   FUNCTION

 Pass One    No Errors Detected
            138 Source Lines
```

File EVAL

```
      1 $DEBUG
      2       SUBROUTINE EVAL(OBJ, X, G, NTERM)
      3 C
      4 C This subroutine evaluates the objective function, OBJ and the
      5 C optimization constraints G
      6 C
      7       REAL L2ERF, G(20), X(40), XE(20), XTAU(20)
      8       INTEGER NTERM
      9 C
     10 C To find the centroidal time for the DOT generated coefficients,
     11 C first have to arrange the X values into coefficients and taus
     12 C so that functions CENT and the error function, L2ERF, will
     13 C understand them.
     14 C
     15       DO 10 I = 1, NTERM
1    16       XE(I) = X(I * 2)
1    17   10    XTAU(I) = X(I * 2 + 1)
     18 C
     19 C Evaluate the objective function
     20 C
     21       OBJ = L2ERF(NTERM, X(1), XE, XTAU)
     22
     23       RETURN
     24       END

Name    Type         Offset P Class

G       REAL              8 *
I       INTEGER*4       176
L2ERF   REAL                FUNCTION
NTERM   INTEGER*4        12 *
OBJ     REAL              0 *
X       REAL              4 *
XE      REAL             16
XTAU    REAL             96


     25 C
     26       REAL FUNCTION L2ERF(M, EO, E, TAU)
     27 C
     28 C This function evaluates the L2 norm error function as the
     29 C DOT objective function
     30 C
     31       COMMON/DATA/ N, NTERM, EDATA(200), TDATA(200)
     32       REAL DIFF, SUM, E(20), TAU(20), EO
     33       SUM = 0.0
     34
     35       DO 10 I=1,N
1    36         DIFF = EDATA(I) - PRONY(TDATA(I), EO, M, E, TAU)
1    37         DIFF = DIFF / EDATA(I)
1    38    10   SUM = SUM + DIFF**2
     39
     40       L2ERF = 100.0 * (SQRT(SUM) / N)
     41       RETURN
     42       END
```

```
Name     Type         Offset P Class

DIFF     REAL            196
E        REAL              8 *
EDATA    REAL              8   /DATA  /
EO       REAL              4 *
I        INTEGER*4       188
M        INTEGER*4         0 *
N        INTEGER*4         0   /DATA  /
NTERM    INTEGER*4         4   /DATA  /
PRONY    REAL                  FUNCTION
SQRT                           INTRINSIC
SUM      REAL            184
TAU      REAL             12 *
TDATA    REAL            808   /DATA  /


      43 C
      44       FUNCTION PRONY(T, EO, M, E, TAU)
      45 C
      46 C This function calculates the value of array E elements
      47 C using the Prony equation, E, and time values.
      48 C
      49       REAL SUM, T, EO, E(20), TAU(20), div
      50       INTEGER M
      51       SUM = 0.0
      52       DO 10 J = 1, M
1     53 C        WRITE(*,*) J, E(J), T, TAU(J)
1     54         div = t/tau(j)
1     55 C    error trap if t over tau gets really big
1     56         if (div .gt. 11000.0) div=11000.0
1     57    10   SUM = SUM + E(J) * EXP(-1.0 * div)
      58
      59       PRONY = EO + SUM
      60       RETURN
      61       END

Name     Type         Offset P Class

DIV      REAL            212
E        REAL             12 *
EO       REAL              4 *
EXP                           INTRINSIC
J        INTEGER*4       204
M        INTEGER*4         8 *
SUM      REAL            200
T        REAL              0 *
TAU      REAL             16 *


Name     Type          Size   Class

DATA                   1608   COMMON
EVAL                          SUBROUTINE
L2ERF    REAL                 FUNCTION
PRONY    REAL                 FUNCTION

 Pass One    No Errors Detected
             61 Source Lines
```

Sample Input Files

Solid Rocket Propellant Stress Relaxation Data

38

0.001	3964.5
0.0023	3823.4
0.0049	3536.3
0.0071	3291.5
0.0097	3111.7
0.0146	2861.3
0.0195	2701.7
0.0264	2537.7
0.0334	2418.4
0.0432	2274.9
0.0555	2159
0.0695	2033.2
0.089	1905.5
0.1126	1793.4
0.1809	1587.5
0.2265	1489.5
0.2938	1397.2
0.3828	1319.6
0.516	1218.1
0.6818	1135
0.9021	1057.5
1.1995	961.6
1.6607	882.4
2.5274	783
3.6455	707.8
4.785	660.4
6.3953	605.8
8.0163	571.5
10.2722	539.4
12.9772	498.3
17.324	478.4
22.284	451.3
28.7547	430.7
36.4954	418
46.2945	391.6
58.3599	362.7
73.6437	357.6
84.5043	361.4

V747-75 Viton Stress Relaxation Data

```
100
0.00000367          5650
0.000004809         5324
0.0000063           4897
0.000008255         4472
0.00001082          4113
0.00001417          3788
0.00001857          3471
0.00002433          3176
0.00003188          2891
0.00004177          2624
0.00005472          2376
0.0000717           2153
0.00009395          1947
0.0001231           1771
0.0001613           1613
0.0002113           1481
0.0002769           1353
0.0003628           1185
0.0004753           1059
0.0006228           940.2
0.000816            856.8
0.001069            789.9
0.001401            734.8
0.001835            687.8
0.002405            644.5
0.003151            601
0.004129            557.5
0.005409            525.7
0.007088            496
0.009287            468.6
0.01217             443.3
0.01594             422.1
0.02089             403.9
0.02737             386.9
0.03586             372.9
0.04699             365
0.06156             357.4
0.08066             347.5
0.1057              333.5
0.1385              321
0.1814              311
0.2377              300.7
0.3115              293.3
0.4081              286
0.5347              276.8
0.7006              272.2
0.918               266.1
1.203               259.9
1.576               253.4
2.065               247.8
2.705               242.5
3.545               237.4
4.645               231.3
6.085               227.5
7.973               222.2
10.45               218
13.69               213.9
```

17.93	209
23.5	204.3
30.79	200
40.34	196
52.86	192.1
69.26	187.7
90.74	186.3
118.9	184.9
155.8	183.1
204.1	178.5
267.4	173.9
350.4	170.8
459.1	167.8
601.5	165
788.2	160.6
1033	158.2
1353	156.7
1773	153.1
2323	151.4
3044	149.5
3988	146.8
5225	145.1
6846	143.3
8970	141.6
11750	139.8
15400	138.1
20180	136.5
26440	136
34640	134.5
45380	133.5
59460	132.3
77910	131.3
102100	129.9
133800	129.1
175200	128.3
229600	127.7
300900	126.6
394200	125.4
516500	124.3
676700	123.2
886700	121.6
1162000	120.5
1522000	119

Sample Output Files

Results for:

Solid Rocket Propellant Stress Relaxation Data

```
number of terms:              6
DOI Results
       360.5667000
       826.2373000         4.874798E-003
      1179.0900000        2.480444E-002
       844.4313000         1.673335E-001
       537.0704000         1.1411860
       304.6428000         7.1204460
       117.0028000        41.5278700

Optimized Result
       353.3835000
       826.2455000         5.623248E-003
      1179.0830000        3.090254E-002
       844.4231000         2.100785E-001
       537.0909000         1.4431110
       304.7081000         9.4107640
       115.8720000        35.2747100
```

Results for:

V747-75 Viton Stress Relaxation Data

```
number of terms:              15
DOI Results
       120.3667000
      3443.7410000            1.109693E-005
      1648.3390000            6.903531E-005
       829.9131000             4.452093E-004
       294.2484000             3.097498E-003
       131.0262000             1.836272E-002
        71.7018400              1.441759E-001
        45.5001500              9.897782E-001
        33.9000300              6.4871400
        27.5999900             39.2573400
        21.3000000            295.8921000
        18.2000000           1726.8700000
        10.8000000          11482.6600000
         7.6999970          80801.2000000
         7.8000030         652020.1000000
         1.5000000        1389563.0000000

Optimized Result
       120.3667000
      3443.7410000            1.109693E-005
      1648.3390000            6.903531E-005
       829.9131000             5.247172E-004
       294.2484000             3.395960E-003
       131.0262000             2.035421E-002
        71.7018400              1.444796E-001
        45.5001500              9.898055E-001
        33.9000300              6.4871430
        27.6000000             39.2573400
        21.3000100            295.8921000
        18.2000000           1726.8700000
        10.8000100          11482.6600000
         7.7000040          80801.2000000
         7.8000090         652020.1000000
         1.5000070        1389563.0000000
```

Sample DOT Output File

```
DDDDD         OOOOO     TTTTTTT
D    D       O     O       T
D     D == O   *   O ==    T
D    D       O     O       T
DDDDD         OOOOO        T
```

DESIGN OPTIMIZATION TOOLS

VERSION 2.04B

- YOUR INTEGRITY IS OUR COPY PROTECTION -

CONTROL PARAMETERS

OPTIMIZATION METHOD, METHOD = 1
NUMBER OF DECISION VARIABLES, NDV = 13
NUMBER OF CONSTRAINTS, NCON = 0
PRINT CONTROL PARAMETER, IPRINT = 3
GRADIENT PARAMETER, IGRAD = 0
GRADIENTS ARE CALCULATED BY DOT
THE OBJECTIVE FUNCTION WILL BE MINIMIZED

-- SCALAR PROGRAM PARAMETERS

REAL PARAMETERS

1) CT = -3.00000E-02	8) DX2 = 2.35818E+02
2) CTMIN = 5.00000E-03	9) FDCH = 1.00000E-03
3) DABOBJ = 1.00000E-04	10) FDCHM = 1.00000E-04
4) DELOBJ = 1.00000E-06	11) RMVLMZ = 4.00000E-01
5) DOBJ1 = 1.00000E-01	12) DABSTR = 7.15109E-04
6) DOBJ2 = 2.00000E-01	13) DELSTR = 1.00000E-03
7) DX1 = 1.00000E-02	

INTEGER PARAMETERS

```
 1) IGRAD =  0    6) NCOLA =  1    11) IPRNT1 =  0
 2) ISCAL = -1    7) IGMAX =  0    12) IPRNT2 =  0
 3) ITMAX = 1000  8) JTMAX = 20    13) JWRITE =  0
 4) ITRMOP = 10   9) ITRMST =  2
 5) IWRITE =  6  10) JPRINT =  0


  STORAGE REQUIREMENTS
ARRAY  DIMENSION  REQUIRED
 WK     1400       365
IWK      200        84


-- INITIAL VARIABLES AND BOUNDS

LOWER BOUNDS ON THE DECISION VARIABLES (XL-VECTOR)
  1)  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
  6)  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10  1.00000E-10
 11)  1.00000E-10  1.00000E-10  1.00000E-10

DECISION VARIABLES (X-VECTOR)
  1)  3.60567E+02  8.26237E+02  4.87480E-03  1.17909E+03  2.48044E-02
  6)  8.44431E+02  1.67334E-01  5.37070E+02  1.14119E+00  3.04643E+02
 11)  7.12045E+00  1.17003E+02  4.15279E+01

UPPER BOUNDS ON THE DECISION VARIABLES (XU-VECTOR)
  1)  1.00000E+20  1.00000E+20  1.00000E+20  1.00000E+20  1.00000E+20
  6)  1.00000E+20  1.00000E+20  1.00000E+20  1.00000E+20  1.00000E+20
 11)  1.00000E+20  1.00000E+20  1.00000E+20


-- INITIAL FUNCTION VALUES

OBJ =  .71511


--- BEGIN UNCONSTRAINED OPTIMIZATION: BFGS METHOD


-- BEGIN ITERATION  1

OBJECTIVE = 6.47015E-01

DECISION VARIABLES (X-VECTOR)
  1)  3.60567E+02  8.26237E+02  7.89760E-03  1.17909E+03  2.79513E-02
  6)  8.44431E+02  1.68008E-01  5.37070E+02  1.14128E+00  3.04643E+02
 11)  7.12046E+00  1.17003E+02  4.15279E+01


-- BEGIN ITERATION  2

OBJECTIVE = 5.89854E-01
```

DECISION VARIABLES (X-VECTOR)
1) 3.60567E+02 8.26237E+02 4.82797E-03 1.17909E+03 3.43469E-02
6) 8.44431E+02 1.70867E-01 5.37070E+02 1.14177E+00 3.04643E+02
11) 7.12051E+00 1.17003E+02 4.15279E+01

-- BEGIN ITERATION 3

OBJECTIVE = 4.77393E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60567E+02 8.26237E+02 6.51326E-03 1.17909E+03 2.91066E-02
6) 8.44431E+02 2.32370E-01 5.37070E+02 1.15610E+00 3.04643E+02
11) 7.12209E+00 1.17003E+02 4.15278E+01

-- BEGIN ITERATION 4

OBJECTIVE = 3.61900E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60568E+02 8.26237E+02 5.16969E-03 1.17909E+03 3.26994E-02
6) 8.44431E+02 2.11180E-01 5.37071E+02 1.51215E+00 3.04645E+02
11) 7.18257E+00 1.17007E+02 4.15262E+01

-- BEGIN ITERATION 5

OBJECTIVE = 3.44395E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60565E+02 8.26237E+02 5.63150E-03 1.17909E+03 3.08455E-02
6) 8.44431E+02 2.00236E-01 5.37071E+02 1.55682E+00 3.04646E+02
11) 7.22203E+00 1.17008E+02 4.15233E+01

-- BEGIN ITERATION 6

OBJECTIVE = 3.40292E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60557E+02 8.26237E+02 5.20600E-03 1.17909E+03 3.05333E-02
6) 8.44431E+02 2.00444E-01 5.37070E+02 1.51728E+00 3.04647E+02
11) 7.30687E+00 1.17010E+02 4.15152E+01

-- BEGIN ITERATION 7

OBJECTIVE = 3.27804E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60522E+02 8.26238E+02 5.62546E-03 1.17909E+03 3.04789E-02
6) 8.44431E+02 2.01980E-01 5.37070E+02 1.41528E+00 3.04652E+02
11) 7.67751E+00 1.17021E+02 4.14789E+01

-- BEGIN ITERATION 8

OBJECTIVE = 3.13886E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60472E+02 8.26238E+02 5.53346E-03 1.17909E+03 3.04801E-02
6) 8.44431E+02 2.05381E-01 5.37069E+02 1.44165E+00 3.04660E+02
11) 8.14683E+00 1.17033E+02 4.14275E+01

-- BEGIN ITERATION 9

OBJECTIVE = 3.13683E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60461E+02 8.26238E+02 5.47618E-03 1.17909E+03 3.07724E-02
6) 8.44431E+02 2.03759E-01 5.37069E+02 1.44224E+00 3.04660E+02
11) 8.16004E+00 1.17032E+02 4.14178E+01

-- BEGIN ITERATION 10

OBJECTIVE = 3.10235E-01

DECISION VARIABLES (X-VECTOR)
1) 3.60090E+02 8.26238E+02 5.39597E-03 1.17909E+03 3.07160E-02
6) 8.44431E+02 2.04729E-01 5.37070E+02 1.38950E+00 3.04664E+02
11) 8.34823E+00 1.16976E+02 4.10943E+01

-- BEGIN ITERATION 11

OBJECTIVE = 2.44468E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53529E+02 8.26245E+02 5.54196E-03 1.17908E+03 3.14716E-02
6) 8.44423E+02 2.02240E-01 5.37090E+02 1.42370E+00 3.04706E+02
11) 9.27941E+00 1.15890E+02 3.53985E+01

-- BEGIN ITERATION 12

OBJECTIVE = 2.37524E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53302E+02 8.26245E+02 5.59169E-03 1.17908E+03 3.10226E-02
6) 8.44423E+02 2.10460E-01 5.37091E+02 1.44385E+00 3.04708E+02
11) 9.37990E+00 1.15856E+02 3.52027E+01

-- BEGIN ITERATION 13

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 14

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 15

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 16

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 17

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 18

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02

6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 19

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 20

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 21

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 22

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

-- BEGIN ITERATION 23

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)
1) 3.53384E+02 8.26245E+02 5.62325E-03 1.17908E+03 3.09025E-02
6) 8.44423E+02 2.10079E-01 5.37091E+02 1.44311E+00 3.04708E+02
11) 9.41076E+00 1.15872E+02 3.52747E+01

— OPTIMIZATION RESULTS

OBJECTIVE = 2.37363E-01

DECISION VARIABLES (X-VECTOR)

1)	3.53384E+02	8.26245E+02	5.62325E-03	1.17908E+03	3.09025E-02
6)	8.44423E+02	2.10079E-01	5.37091E+02	1.44311E+00	3.04708E+02
11)	9.41076E+00	1.15872E+02	3.52747E+01		

TERMINATION CRITERIA

RELATIVE CONVERGENCE CRITERION WAS MET FOR 10 CONSECUTIVE ITERATIONS

ABSOLUTE CONVERGENCE CRITERION WAS MET FOR 10 CONSECUTIVE ITERATIONS

– OPTIMIZATION RESULTS

OBJECTIVE, F(X) = 2.37363E-01

DECISION VARIABLES, X

ID	XL	X	XU
1	1.00000E-10	3.53384E+02	1.00000E+20
2	1.00000E-10	8.26245E+02	1.00000E+20
3	1.00000E-10	5.62325E-03	1.00000E+20
4	1.00000E-10	1.17908E+03	1.00000E+20
5	1.00000E-10	3.09025E-02	1.00000E+20
6	1.00000E-10	8.44423E+02	1.00000E+20
7	1.00000E-10	2.10079E-01	1.00000E+20
8	1.00000E-10	5.37091E+02	1.00000E+20
9	1.00000E-10	1.44311E+00	1.00000E+20
10	1.00000E-10	3.04708E+02	1.00000E+20
11	1.00000E-10	9.41076E+00	1.00000E+20
12	1.00000E-10	1.15872E+02	1.00000E+20
13	1.00000E-10	3.52747E+01	1.00000E+20

FUNCTION CALLS = 271